

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	08
1.	INTRODUCTION	10
2.	SCOPE OF THE PROJECT	12
3.	LITERATURE SURVEY	13
4.	REQUIREMENT SPECIFICATION	16
5.	SYSTEM DESIGN	18
6.	MODULES	25
7.	MODULE DESCRIPTION	32
8.	SYSTEM TESTING	37
9.	CODING	40
10.	SCREENSHOTS	44
11.	CONCLUSION	48
12.	APPENDICES	49
13.	REFERENCES	50

ABSTRACT

In many educational institutions, students often face the challenge of managing their expenses for essential items like food and stationery within the campus. To streamline this process and offer a more convenient payment system, this project proposes an "Id Barcode Payment System." The system allows students to use a personalized barcode for purchasing items from the college canteen, stationery store, or any on-campus retail outlets. Instead of carrying cash, students can scan their unique barcode at the point of sale, and the amount spent is directly added to their student account or tuition fees. This ensures a cashless, secure, and efficient transaction process, reducing the need for students to carry money while offering easy tracking of expenses. The system can be integrated into the institution's existing database, ensuring smooth billing, easy management, and increased transparency for both students and administration.

LIST OF FIGURES

FIGURE NO.	NAME OF THE FIGURE	PAGE NO.
5.1.1	Architecture Diagram	19
5.2.1	Entity Relationship Diagram	21
5.3.1	Data Flow Diagram	22
5.4.1	Use case Diagram	24
6.1.1	Front-end Modules	26
6.2.1	Back-end Modules	28
6.3.1	Database Modules	30
6.4.1	Additional Utility Modules	31

CHAPTER 1

INTRODUCTION

This project's initial idea is an innovative solution designed to address the challenges faced by students in managing their daily expenses within educational institutions. Many students often need to buy items such as food, beverages, and stationery from on-campus outlets, but the lack of cash can be a hindrance, especially when digital payment options are limited. To bridge this gap, the ID Barcode Payment System offers a simple, secure, and efficient way for students to pay for their purchases without needing physical money or cards.

The core idea behind this system is to provide students with a personalized barcode linked to their student account. This barcode can be scanned at various on-campus points of sale, such as the cafeteria, library, or any retail outlet within the institution. Each time a student scans their barcode for a purchase, the transaction amount is automatically added to their account, which may be settled at the end of a specified period, typically with the student's tuition fees. This enables a seamless, cashless transaction process that is not only convenient for students but also helps the institution manage payments in a streamlined manner.

PROBLEM STATEMENT

In many educational institutions, managing student expenses for on-campus purchases such as food, stationery, and other essentials can be challenging. Students often face inconvenience when they do not have cash on hand, and limited digital payment options can further complicate transactions. This leads to inefficiencies, longer queues at payment counters, and increased administrative work in handling cash transactions. Moreover, the lack of transparency in tracking individual expenses makes it difficult for students and their families to monitor spending. To address these challenges, our project aims to provide a cashless, secure, and efficient solution. By allowing students to scan their personalized barcodes for purchases, the system automatically adds the expense to their student account or fee structure, ensuring seamless transactions and accurate record-keeping. This reduces the reliance on cash, simplifies expense tracking, and integrates smoothly with existing financial management systems of the institution.

CHAPTER 2

SCOPE OF THE PROJECT

This project is designed to streamline student payments through a cashless solution. Developed using Python and Flask for the backend, and Electron with JavaScript for the frontend, this system allows students to scan their ID barcodes to manage expenses directly through their fee account when they lack adequate cash. It integrates seamlessly with existing fee management systems, automatically updating the student's balance whenever a transaction is made, such as for library fees, cafeteria bills, or event charges. The system offers a cross-platform solution, ensuring compatibility with Windows, macOS, and Linux. It emphasizes user authentication and security by verifying student identity through barcode scanning and protecting sensitive data with secure practices. Additionally, it provides detailed reports on individual and collective student expenses, enabling administrators to track payments, dues, and transaction history efficiently. Initially intended for colleges, the system is also adaptable for use in schools and offices, demonstrating its flexibility across different environments. Its user-friendly interface makes it easy for both students and administrators to use, while the automation of payment processes reduces the time and cost associated with manual handling. Overall, the Campus Barcode Payment System aims to offer a secure, efficient, and scalable solution for managing student expenses.

CHAPTER 3

LITERATURE SURVEY

3.1 NAME OF THE PAPER: The Logistics Barcode ID Character Recognition Method Based on AKAZE Feature Localization

YEAR OF PUBLICATION: 2022

INFERENCE: With the continuous development of science and technology, the rapid rise of e-commerce, and the continuous influx of the logistics industry into our lives. In the transportation of logistics, the sorting is the mainly work, it needs machines or staff using barcode scanners to identify barcodes to achieve sorting. Meantime, it is necessary to align the barcode of the item manually or mechanically with the barcode scanner to accurately identify it. Undoubtedly, a lot of manpower and material resources are required in the sorting process, so the method of computer vision processing is used to achieve automatic detection and identification of barcode ID will have great significance. The authors propose a logistics barcode ID character recognition method based on local positioning of AKAZE features to realize the detection and identification of logistics barcode ID in a long distance and a large range. First, extracting the AKAZE feature of the barcode in the image, using the matching algorithm to localize the barcode area, and then use the OCR (Optical Character Recognition) method based on deep learning to perform the character recognition of the barcode ID. A large number of experiments have been carried out with the real existing parcel barcodes, in which the accuracy rate reaches 99.6%, the speed reaches 0.35s/frame. Besides, the detection and identification of barcode IDs of different types and different length codes can be realized. The adoption in the future will further promote the feasibility of the logistics and transportation industry.

3.2 NAME OF THE PAPER: Application of Smart Phone for Industrial Barcode Scanner.

YEAR OF PUBLICATION: 2021

INFERENCE: The barcode system is used in various purpose. In industrial sector, barcode is used for identifying product in the manufacturing process. However, the barcode systems in industrial factory requires special barcode reader and the industrial barcode reader is more complicated and expensive than the conventional barcode reader. Recently, all smart phone can

be applied for barcode reader application but there is no study how to use the smart phone barcode reader application in industrial factory. Therefore, this study would consider how to use the smart phone for industrial barcode reader system. The Android smart phone was used to develop barcode reader application by cooperating with the barcode reader library and the barcode reader system in manufacturing process was simulated. The barcodes for this experiment were ID barcode and QR code. Moreover, the damaged barcodes were tested in the experiments and each experimental was continue tested nonstop for 24 hours. The result found that smart phone can read ID barcode and QR code. The readable rates of ID barcode (complete and damage barcode) were 100%. The readable rates of QR code at 30 and 50 photos per minute were 92.5% and 85%. Thus, the smart phone has an opportunity to use as ID barcode reader in industrial factory. For QR code, the readable rate result is low. However, there are many advantages to use smart phone for barcode reader such as price, flexibility, easy to use and reinstall. So, it is a challenge to use smart phone for industrial barcode reader.

3.3 NAME OF THE PAPER: Design of bar code automatic identification system based on mobile terminal.

YEAR OF PUBLICATION: 2020

INFERENCE: This paper addresses the critical issue of automatic barcode identification within the context of agricultural product traceability, a process essential for safeguarding the integrity of special agricultural products. Effective traceability not only enhances product safety but also plays a vital role in poverty alleviation and increasing farmers' incomes. The study begins with an in-depth analysis of the EAN/UCC-128 barcode system, highlighting its advantages in ensuring accurate tracking and efficient supply chain management. Furthermore, it explores the characteristics of mobile terminals utilized in barcode recognition, focusing on their ability to facilitate rapid scanning and real-time data processing. Research findings demonstrate that optimizing barcode scanning can lead to significant reductions in both storage space and identification time, making this automatic recognition system highly applicable in agricultural settings. By leveraging technology for improved traceability, this approach promises to enhance market access for farmers and boost consumer confidence in agricultural products. Ultimately, this paper underscores the transformative potential of automated identification systems in fostering economic growth and sustainability in the agricultural sector.

3.4 NAME OF THE PAPER: Design of a barcode identification system

YEAR OF PUBLICATION: 2014

INFERENCE: Modern barcode readers need to decode many symbology's. It is inefficient for the readers to read and decode a barcode symbol in the try-and-error manner. That is, it is inefficient to decode a given symbol by testing all symbology's supported by the reader. A more efficient decoding scheme is to identify the symbology used by the target symbol. Then the decoder in the reader uses the correct symbology for decoding. In this paper, we propose a method of identifying the QR and Aztec barcodes. We use the connected component labelling algorithm to compute the tag for each connected component, then search for the innermost connected component of the finder patterns in the tag image. Experimental results show that the method can identify rotated or defocused barcode images.

3.5 NAME OF THE PAPER: A mobile payment scheme using 2D-barcode.

YEAR OF PUBLICATION: 2013

INFERENCE: The mobile payment service is an important and popular service that attracts attentions. Number of mobile users is growing up every day. The users like to have more and more services ready on the wireless networks. Companies try to produce more smart phones while customers buy and use them. More smart phones mean a more proper infrastructure to use 2-dimensional barcodes (2D-barcodes) in mobile payment solutions. We have designed and developed a new m-payment approach in which customers use their mobile phones as 2D-barcode encoder to generate 2D-barcodes. Then, they send the generated 2D-barcodes to the merchant's terminal as valuable coupons. The method introduced here is easy for customer to utilize and is light for mobile set to process; it also does not impose a high cost for customers when they want to sign up.

CHAPTER 4

REQUIREMENT SPECIFICATION

The Barcode Management System is designed to facilitate cashless transactions in offices, colleges, and schools by utilizing the barcode present on ID cards. This system allows users to scan their ID card's barcode when making a purchase, and the corresponding amount is added to their account. This offers flexibility, enabling users to complete transactions even if they do not have money at the time, with the option to settle the charges later.

Below is a detailed requirement specification for the system:

4.1. Functional Requirements

User Authentication: The system should authenticate users by scanning the barcode present on their ID card. Each ID barcode is linked to the user's account, ensuring that transactions are accurately tracked.

Barcode Scanning: The system must support barcode scanning using the user's ID card. This will trigger the system to record the transaction and update the user's account with the amount owed.

Transaction Management: Once a barcode is scanned, the system must add the purchase amount to the user's balance. This transaction is recorded in the database for later settlement when the user pays the amount due.

Account Management: Users should be able to view their outstanding balance and pay it at a later date. The system should generate a detailed transaction history for each user.

Database Management: A centralized database must store user details, transaction history, and account balances. It should be regularly updated and accessible in real-time for accurate processing.

4.2. Non-Functional Requirements

Usability: The system should be intuitive, allowing users to quickly scan their ID card and complete transactions without difficulty.

Performance: The system should efficiently process multiple barcode scans and transaction updates in real-time without lag, ensuring smooth operation.

Security: User account data and transaction details must be securely encrypted to protect sensitive information from unauthorized access.

Scalability: The system should be capable of handling a large number of users and transactions, and adaptable to additional environments like cafeterias, libraries, and other services within the institution.

4.3. Technical Requirements

Frontend: Built using Electron, HTML, and JavaScript to create a responsive, user-friendly interface.

Backend: Developed with Python Flask to manage barcode scanning, transaction processing, and database communication.

Database: Utilizes a relational database such as MySQL or PostgreSQL to store user account data, transaction history, and balances.

Barcode Scanner: The system must integrate with barcode scanning devices to scan ID cards for accurate user identification and transaction management.

In conclusion, the requirement specification for the barcode to PC server application utilizing Electron, Node.js, and Python Flask provides a comprehensive framework for developing a robust and efficient system. By leveraging Electron, we ensure a cross-platform desktop application that delivers a seamless user experience, while Node.js facilitates real-time data processing and communication with the server. Python Flask serves as a lightweight yet powerful backend, handling data management and API interactions effectively. Additionally, the hardware requirements outlined ensure optimal performance and reliability, enabling smooth integration with barcode scanning devices and enhancing overall functionality. This specification lays a solid foundation for the development process, aligning technical capabilities with user needs and project goals.

CHAPTER 5

SYSTEM DESIGN

5.1 ARCHITECTURE DIAGRAM:

An architecture diagram visually represents the structure and components of a system, illustrating how various elements interact with each other. It serves as a blueprint, detailing the relationships between hardware, software, and network components. This diagram aids stakeholders in understanding the system's design and functionality, facilitating communication among developers, architects, and clients. By highlighting key components and their interactions, it helps identify potential challenges and areas for improvement. Overall, architecture diagrams are essential for guiding development and ensuring alignment with project objectives.

The architecture of the "ID Barcode Software" integrates Python Flask and Electron to create a powerful desktop application. At the backend, Flask serves as the API server, responsible for handling business logic and processing requests. It interacts with a database, such as PostgreSQL or SQLite, to store user ID information and barcode data. Flask's RESTful API provides endpoints for creating, reading, updating, and deleting records, ensuring efficient data management. On the client side, Electron is utilized to build a cross-platform desktop application that offers a responsive user interface. The Electron app communicates with the Flask API to retrieve ID data and generate barcodes on-demand. To enhance performance, the application can implement local caching, allowing users to access previously loaded data without constantly hitting the server. Additionally, Electron's capabilities enable barcode scanning through integrated camera features or external devices. This architecture ensures real-time synchronization between the backend and frontend while providing a smooth user experience. Security measures, such as API authentication and encryption, are crucial to protect sensitive user data. Overall, this architecture combines the strengths of Flask and Electron to deliver a robust and efficient solution for managing ID barcodes.

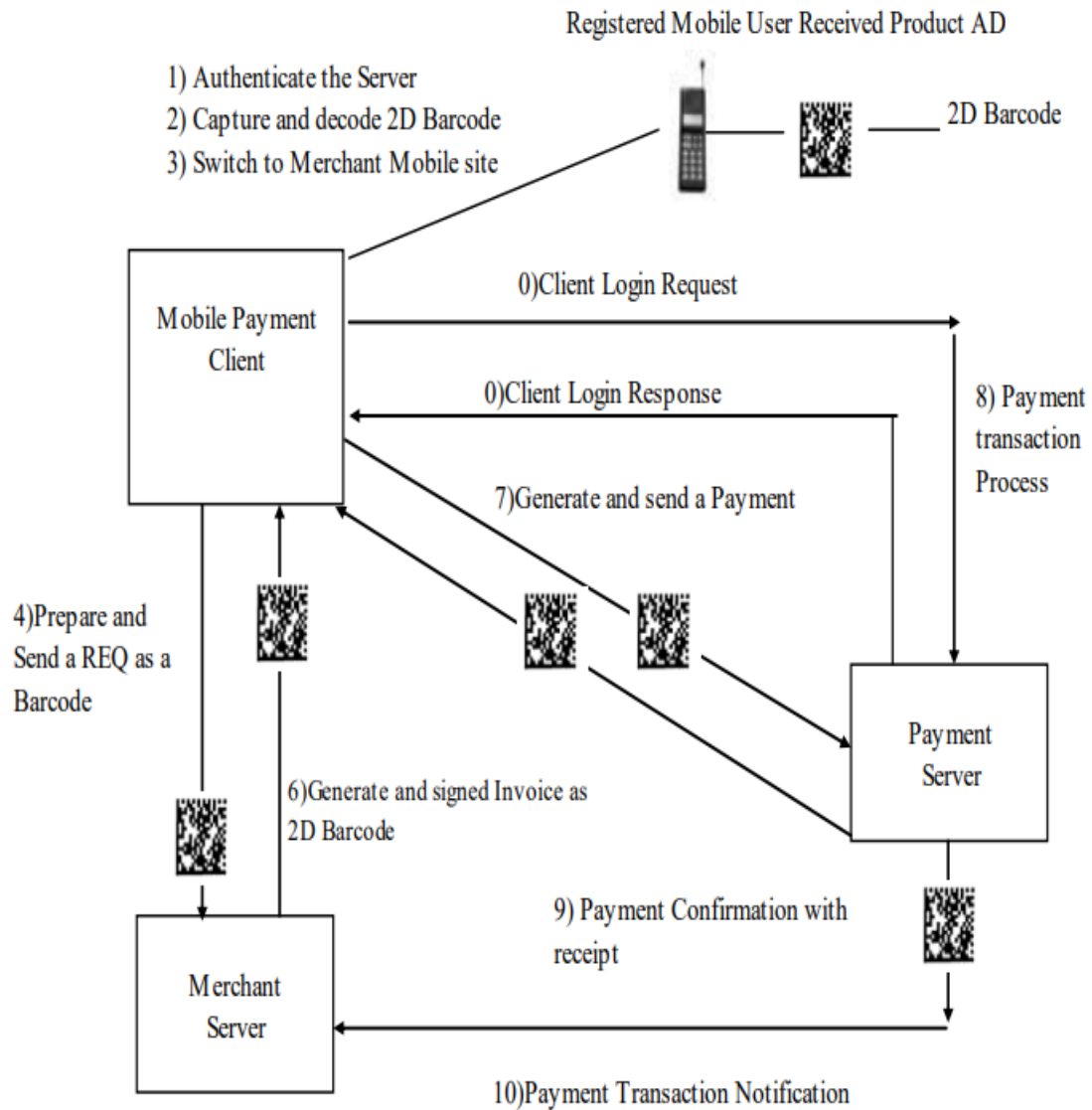


FIG 5.1.1: ID BARCODE PAYMENT SYSTEM ARCHITECTURE DIAGRAM

5.2 ENTITY - RELATIONSHIP DIAGRAMS:

ENTITIES:

The “ID Barcode Payment System” entity is designed to manage transactions initiated via barcodes, encapsulating essential details for effective tracking and processing. Entities are analogous to relations in the relational model. They represent the principle data objects about which information is to be collected. Entities represent concepts or concrete or abstract objects such as person, place, physical things, events. In an E-R diagram, an entity is represented as a named rectangular shape, which may include a list of attributes. For clarity, normally only attributes that are involved in relationships between entities are included, i.e. primary key (PK) and foreign keys (FK). This maintains an uncluttered diagram.

RELATIONSHIP:

A relationship represents some association between two or more entities. In an E-R diagram, a relationship is represented as a diamond shape, containing the name of the relationship between the entities. A relationship in the context of a project refers to the connections and interactions between various entities involved, such as team members, stakeholders, and external partners. These relationships can be defined by communication patterns, collaboration dynamics, and the flow of information. Effective relationships are crucial for project success, as they foster trust, enhance teamwork, and facilitate the alignment of goals. Understanding these relationships allows for better coordination, conflict resolution, and resource management, ultimately contributing to the achievement of project objectives. Nurturing positive relationships also promotes a collaborative environment, encouraging innovation and adaptability in the face of challenges.

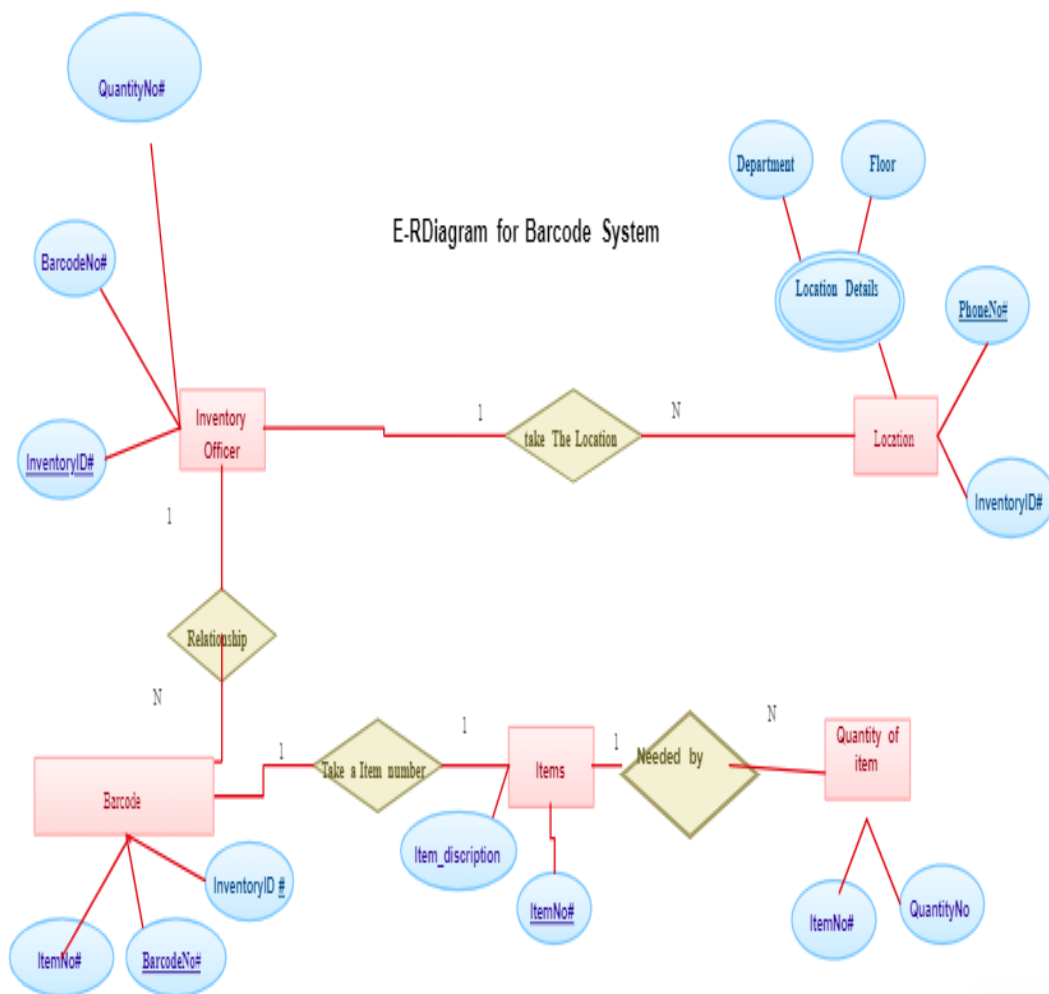


FIG 5.2.1: ID BARCODE PAYMENT SYSTEM E-R DIAGRAM

5.3 DATA FLOW DIAGRAM:

A Data Flow Diagram (DFD) is a visual representation of the flow of data within a system, illustrating how information is processed and transferred. It consists of key components such as processes, data flows, data stores, and external entities, which collectively depict the interactions and transformations of data. DFDs are useful for analysing and designing systems, providing a clear overview of data movement and processing. They help stakeholders understand workflows and identify potential areas for improvement. By varying levels of detail, DFDs can range from high-level context diagrams to more granular representations of specific processes.

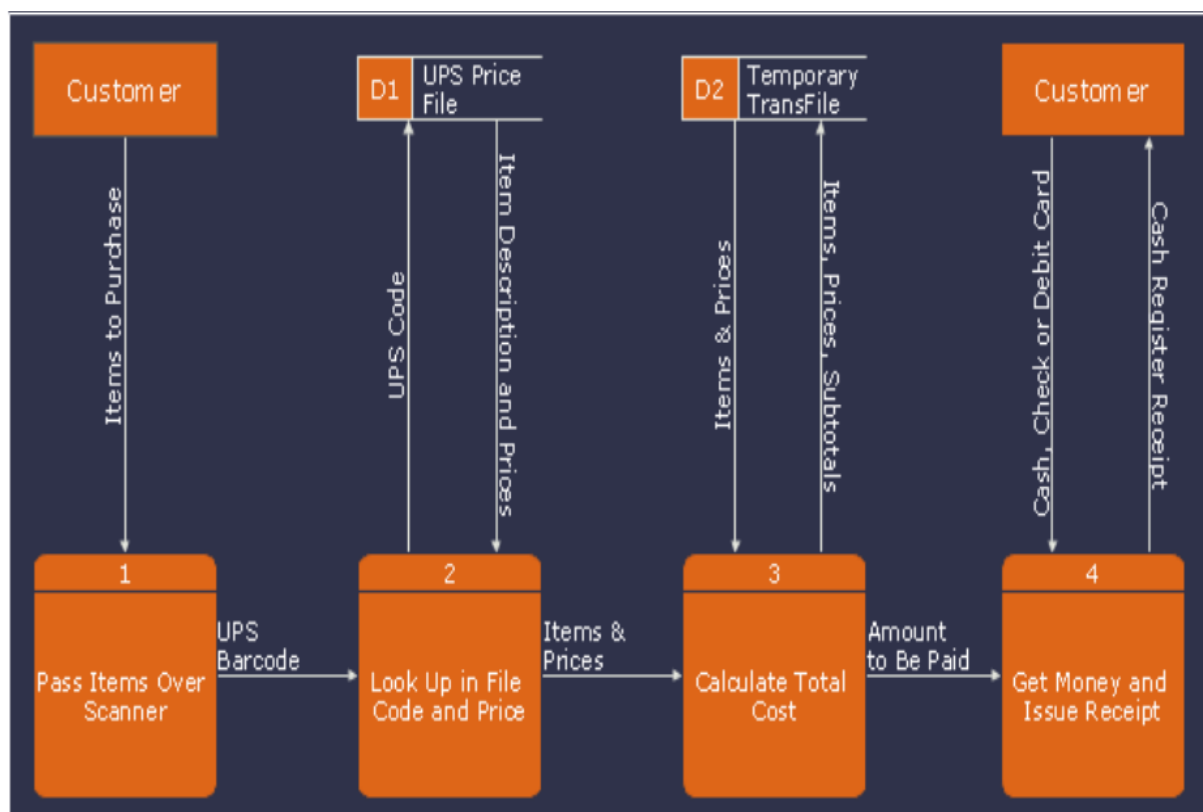


FIG 5.3.1: ID BARCODE PAYMENT SYSTEM DATA FLOW DIAGRAM

5.4 USE CASE DIAGRAM:

The purpose of the use case diagram is to capture the dynamic aspect of a system. However, this definition is too generic to describe the purpose, as other four diagrams (activity, sequence, collaboration, and State chart) also have the same purpose. We will look into some specific purpose, which will distinguish it from other four diagrams.

Use-case diagrams are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements. Hence, when a system is analysed to gather its functionalities, use cases are prepared and actors are identified. When the initial task is complete, use case diagrams are modelled to present the outside view.

In brief, the purposes of use case diagrams can be said to be as follows –

1. **Visual Representation:** Use case diagrams provide a clear visual representation of system functionality, making it easier to understand how users interact with the system.
2. **Stakeholder Communication:** They facilitate communication among stakeholders, including developers, clients, and end-users, by illustrating requirements in a straightforward manner.
3. **Requirement Gathering:** Use case diagrams help in identifying and gathering functional requirements by highlighting user needs and the interactions required to meet those needs.
4. **Scope Definition:** They aid in defining the scope of the system by delineating what is included (and excluded) in the project, helping to manage expectations.
5. **System Boundaries:** Use case diagrams clarify system boundaries by showing what is inside and outside the system, which helps in understanding interactions with external entities.
6. **Documentation:** They serve as a valuable part of system documentation, providing a high-level overview that can be referenced throughout the project lifecycle.
7. **Validation and Verification:** Use case diagrams assist in validating and verifying requirements by allowing stakeholders to review and confirm that all necessary interactions are captured and understood.

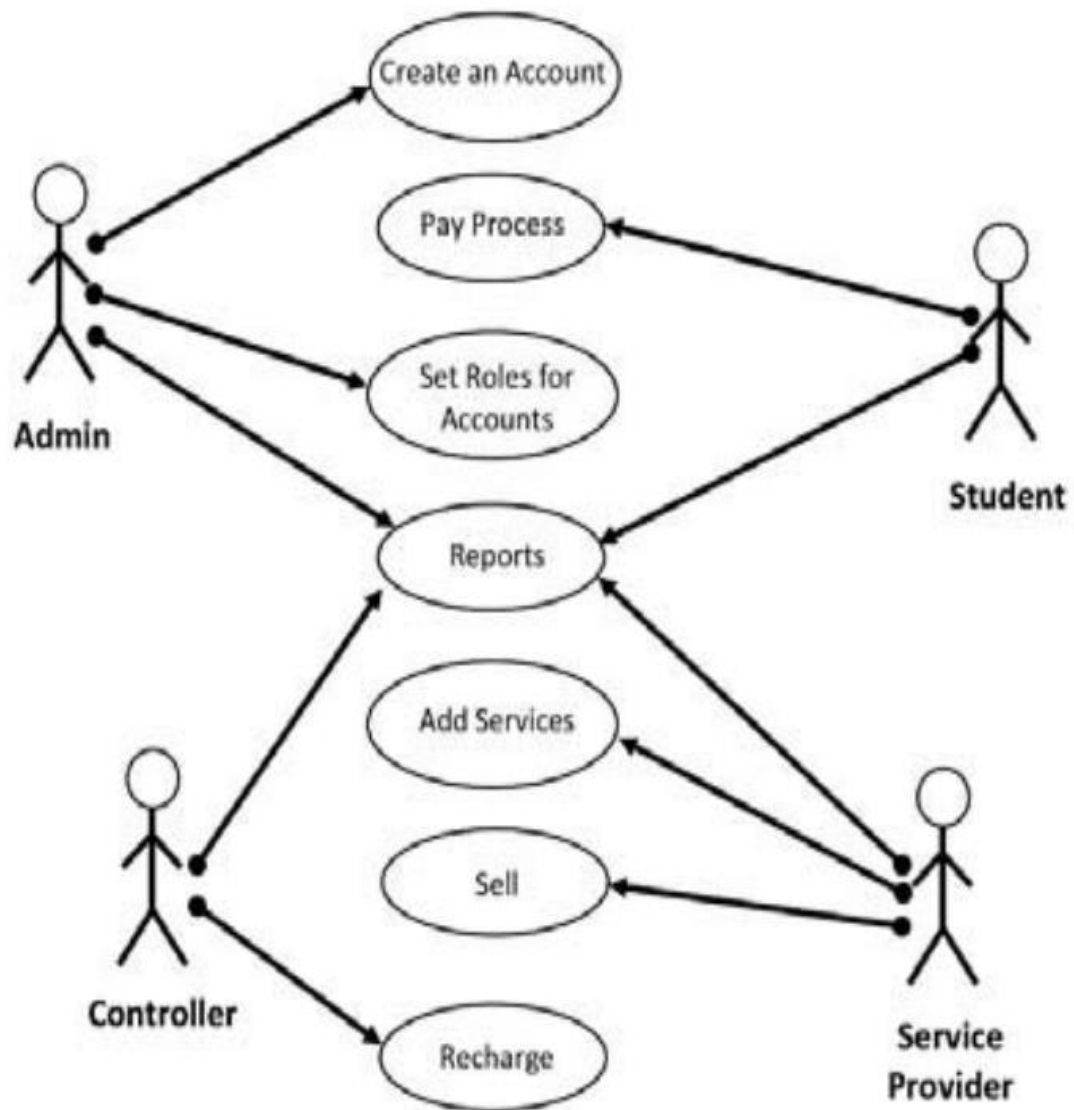


FIG 5.4.1: ID BARCODE PAYMENT SYSTEM USE CASE DIAGRAM

CHAPTER 6

MODULES

MODULE 6.1: FRONTEND MODULES (ELECTRON, JAVASCRIPT (NODE.JS))

- 6.1.1 User Interface (UI) Module
 - Purpose: To provide an intuitive and easy-to-use interface for users to scan their ID barcode.
 - Responsibilities:
 - Display input fields and buttons for barcode scanning.
 - Show user account details (e.g., outstanding balance, transaction history).
 - Provide a summary of the user's total charges after scanning.
 - Responsive design to fit different environments (office, school, college).
- 6.1.2 Barcode Scanning Module
 - Purpose: Handle real-time scanning of barcodes using the user's ID card.
 - Responsibilities:
 - Integrate with hardware barcode scanners.
 - Capture scanned barcode data and send it to the backend via API calls.
 - Display real-time feedback to the user after a successful or failed scan.
- 6.1.3 Transaction Summary and Payment Module
 - Purpose: Display the current balance and transaction summary to the user.
 - Responsibilities:
 - Fetch and display transaction history (retrieved from the backend).
 - Provide options to view pending payments, due dates, and account status.

Electron Architecture

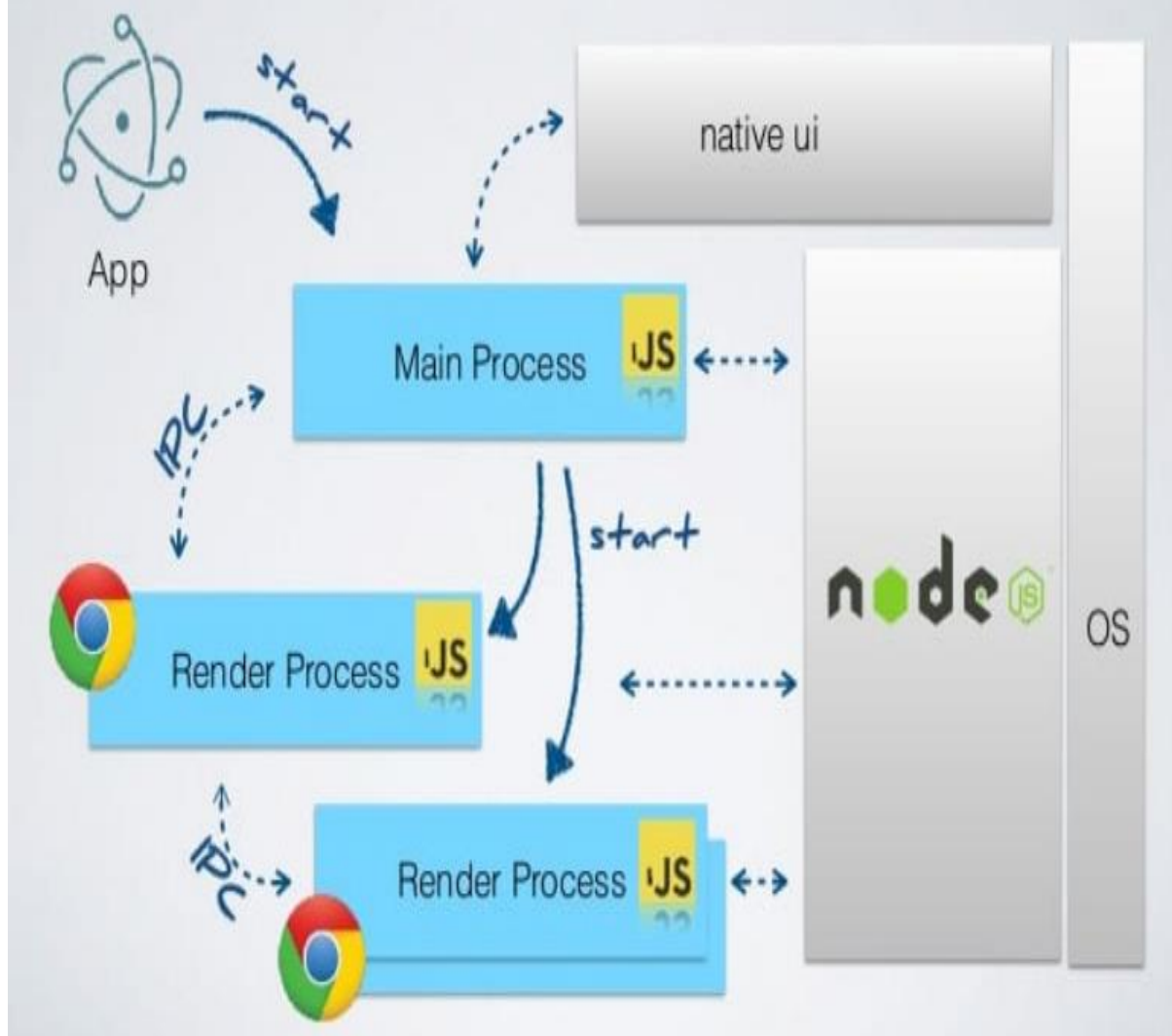


FIG 6.1.1: FRONT-END MODULES (ELECTRON, NODE.JS)

MODULE 6.2: BACKEND MODULES (PYTHON FLASK)

- 6.2.1 User Authentication Module
 - Purpose: Authenticate users based on their ID card's barcode.
 - Responsibilities:
 - Handle user login using barcode data from the frontend.
 - Verify the user's identity from the database and generate session tokens.
 - Ensure secure and authenticated access to transaction and account data.
- 6.2.2 Barcode Processing and Transaction Module
 - Purpose: Process barcode scans and manage transactions.
 - Responsibilities:
 - Validate scanned barcode data and identify the user from the database.
 - Retrieve the corresponding product details (if applicable) and price.
 - Add the amount to the user's balance and update the database with the transaction.
- 6.2.3 Account and Payment Management Module
 - Purpose: Manage user account balances, transactions, and payment status.
 - Responsibilities:
 - Track outstanding balances and allow payment processing (if applicable).
 - Generate detailed transaction histories for users, including timestamps, items purchased, and total charges.
 - Ensure data consistency when updating balances and transaction records.

- 6.2.4 API Management Module
 - Purpose: Facilitate communication between the frontend and backend.
 - Responsibilities:
 - Expose RESTful APIs for barcode scanning, user account retrieval, and transaction updates.
 - Secure API endpoints to ensure authorized access only.

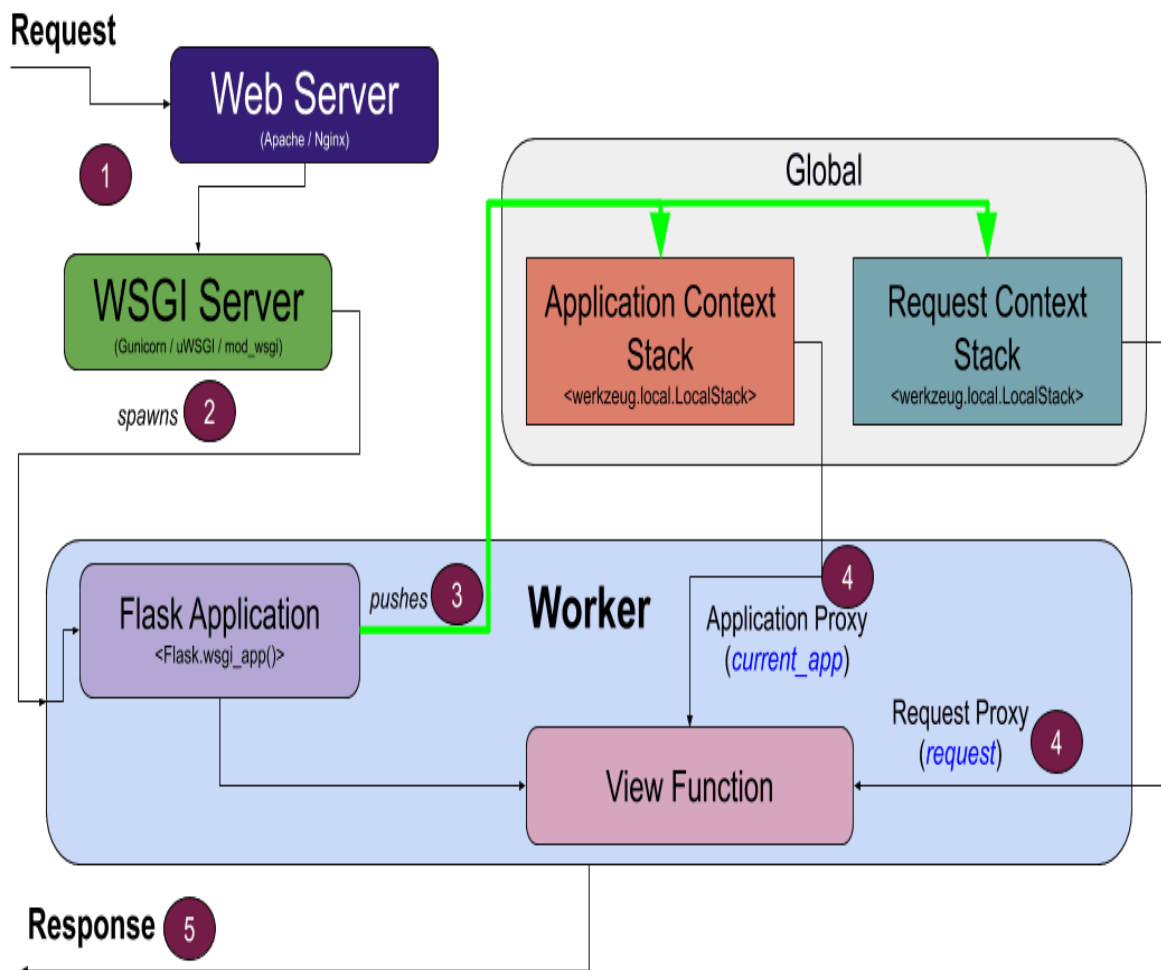


FIG 6.2.1: BACK-END MODULES

MODULE 6.3: DATABASE MODULES (MYSQL/POSTGRESQL)

- 6.3.1 User Data Management Module
 - Purpose: Store and manage user information, such as account details and balances.
 - Responsibilities:
 - Maintain a table for user profiles, linking each user to their unique barcode and account data.
 - Store user balances, transaction history, and payment status.
- 6.3.2 Transaction Data Management Module
 - Purpose: Record and store details of all transactions performed via barcode scanning.
 - Responsibilities:
 - Maintain a transaction history table that logs user purchases.
 - Track the date, time, item purchased, and amount charged for each transaction.
- 6.3.3 Item/Product Data Management Module
 - Purpose: Maintain a list of items or services that can be purchased using the barcode system.
 - Responsibilities:
 - Store information about canteen items, office supplies, or other services available for purchase.
 - Link item prices with barcodes scanned to ensure correct billing during transactions.

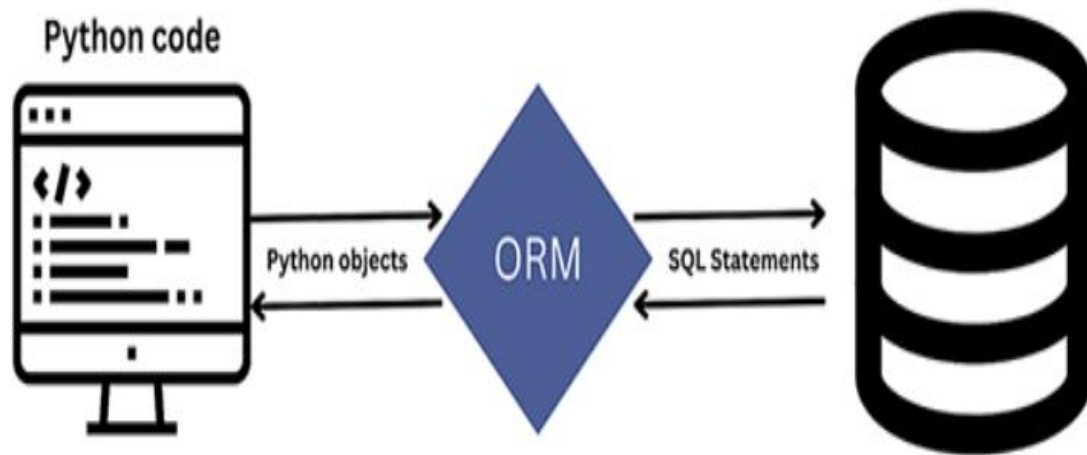


FIG 6.3.1: DATABASE MODULES

MODULE 6.4: ADDITIONAL UTILITY MODULES

- 6.4.1 Security Module
 - Purpose: Ensure data security and protect user credentials and financial information.
 - Responsibilities:
 - Implement encryption for sensitive data such as user credentials and financial transactions.
 - Ensure secure communication between the frontend and backend (e.g., using HTTPS).
- 6.4.2 Error Handling and Logging Module
 - Purpose: Provide detailed logging and handle errors gracefully across the system.

- Responsibilities:
 - Capture and log errors occurring during barcode scans, transaction processing, or database interactions.
 - Provide feedback to the user in case of any issues with scanning or transaction updates.

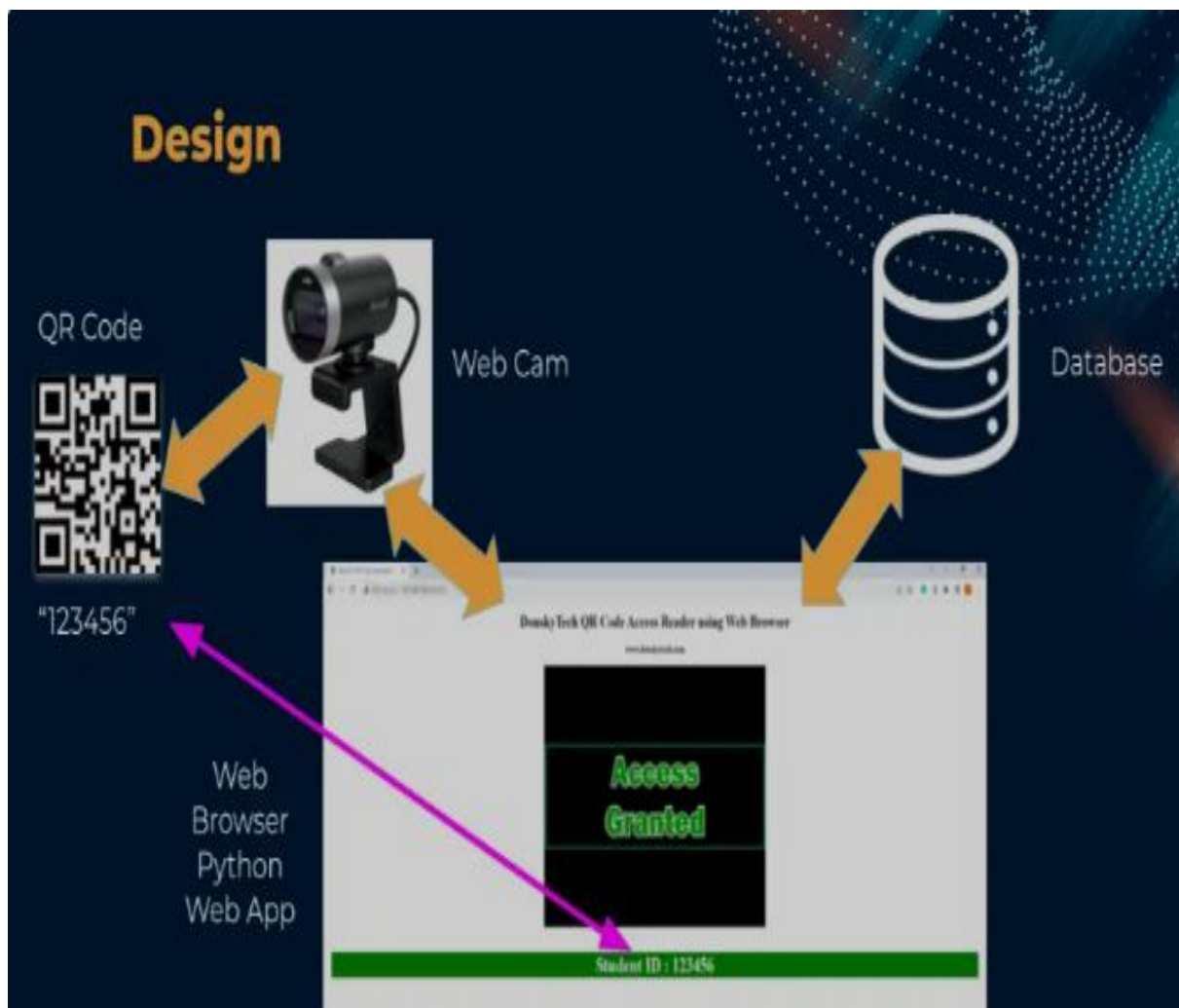


FIG 6.4.1: ADDITIONAL UTILITY MODULES

CHAPTER 7

MODULE DESCRIPTION

7.1. FRONTEND MODULES (ELECTRON, JAVASCRIPT)

- 7.1.1 User Interface (UI)
 - Description: This module is responsible for rendering the user interface where users can interact with the system. It provides a clean, user-friendly layout that is responsive across different devices.
 - Key Features:
 - Display user profile, balance, and transaction history.
 - A clean layout for scanning barcodes.
 - Responsive design to ensure compatibility across multiple devices.
 - Error and success messages after each transaction or scan.
- 7.1.2 Barcode Scanning
 - Description: This module manages the interaction between the user and the barcode scanner. It captures the scanned barcode and sends the data to the backend for processing.
 - Key Features:
 - Integrates with hardware barcode scanners.
 - Captures the scanned barcode data (ID card) and sends it to the backend via API.
 - Provides real-time feedback to the user (e.g., "Scan successful" or "Scan failed").
 - Ensures data validation before sending the request to the backend.

- 7.1.3 Transaction Summary and Payment
 - Description: This module displays the user's current balance and payment summary. It fetches transaction data from the backend and presents it in a readable format for users.
 - Key Features:
 - Retrieves and displays user transaction history.
 - Shows the current outstanding balance (pending payments).
 - Optionally, allows users to view pending payments and payment due dates.
 - Ensures smooth integration with backend APIs to get real-time data.

7.2. BACKEND MODULES (PYTHON FLASK)

- 7.2.1 User Authentication
 - Description: Handles the authentication of users based on the barcode scanned from their ID card. It ensures that only authorized users can access and perform transactions within the system.
 - Key Features:
 - Validates user identity by scanning ID barcode and checking against database records.
 - Manages user sessions and authentication tokens.
 - Securely stores and retrieves user credentials.
- 7.2.2 Barcode Processing and Transaction
 - Description: This core module processes the scanned barcode and manages transactions based on the user's profile. It retrieves relevant item information and adds the cost to the user's account balance.

- Key Features:
 - Processes barcode scans and links them to specific user accounts.
 - Retrieves item data (e.g., price, description) and calculates charges.
 - Updates user balances by adding the transaction amount to the outstanding account.
 - Ensures real-time updates for balance and transactions.
- 7.2.3 Account and Payment Management
 - Description: This module manages the user's account, including their balance, payment history, and outstanding charges. It ensures that users can view their financial details and settle payments later.
 - Key Features:
 - Maintains real-time balance tracking for each user.
 - Provides functionality to update account balances after each transaction.
 - Generates detailed transaction history for each user.
 - Allows users to view their outstanding charges and due payments.
- 7.2.4 API Management
 - Description: This module provides the communication layer between the frontend and backend by exposing RESTful APIs. These APIs handle requests such as barcode scanning, transaction updates, and retrieving user account details.
 - Key Features:
 - Exposes secure endpoints for barcode scanning, user login, and transaction management.
 - Provides real-time data access for frontend modules.
 - Ensures data integrity by validating requests from the frontend.
 - Uses security features like token-based authentication and encryption.

7.3. DATABASE MODULES (MYSQL/POSTGRESQL)

- 7.3.1 User Data Management
 - Description: This module handles the storage and management of user-related information, such as user profiles, account balances, and transaction history.
 - Key Features:
 - Stores user information (e.g., name, student ID, barcode, and balance).
 - Tracks account balances and updates them after each transaction.
 - Ensures data consistency and security during read/write operations.
 - Provides real-time access to user data for backend modules.
- 7.3.2 Transaction Data Management
 - Description: This module records all transactions performed by users. It stores data related to the item purchased, cost, and timestamp, allowing for accurate financial tracking.
 - Key Features:
 - Stores transaction records linked to individual user accounts.
 - Tracks item details (e.g., barcode, price, description) and timestamp of the transaction.
 - Ensures consistency in transaction history for auditing purposes.
 - Provides APIs for accessing and updating transaction data in real time.
- 7.3.3 Item/Product Data Management
 - Description: Manages a catalogue of items or services that can be purchased using the barcode system. Each item has a price associated with it, which is added to the user's account when scanned.
 - Key Features:
 - Maintains a list of products and services that can be purchased.
 - Links barcode IDs with product information for accurate billing.

- Ensures up-to-date prices for items purchased through the system.
- Allows for easy updating and retrieval of product data.

7.4. ADDITIONAL UTILITY MODULES

- **7.4.1 Security Module**

- Description: Ensures that all sensitive user information and financial data are securely handled throughout the system. This module implements encryption, secure communication channels, and access control.
- Key Features:
 - Encrypts sensitive data, such as user credentials and financial transactions.
 - Implements HTTPS for secure data transmission between frontend and backend.
 - Provides authentication and authorization mechanisms for accessing the system.
 - Logs security-related events for monitoring and auditing.

- **7.4.2 Error Handling and Logging**

- Description: This module handles errors and logs them for troubleshooting. It ensures that users receive proper feedback if something goes wrong, and provides detailed logs for developers to diagnose issues.
- Key Features:
 - Logs errors occurring during barcode scanning, database interactions, and API requests.
 - Displays user-friendly error messages in the frontend when issues occur.
 - Captures detailed logs for backend processes to aid in debugging.
 - Tracks system health and reports any critical failures.

CHAPTER 8

SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub – assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

TYPES OF TESTS:

UNIT TESTING:

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. It is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

INTEGRATION TESTING:

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfactory, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

FUNCTIONAL TEST:

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals. Functional testing is centered on the following items:

Valid Input: identified classes of valid input must be accepted.

Invalid Input: identified classes of invalid input must be rejected.

Functions: identified functions must be exercised.

Output: identified classes of application outputs must be exercised.

Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

SYSTEM TESTING:

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

WHITE BOX TESTING:

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

BLACK BOX TESTING:

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box. You cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

UNIT TESTING:

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach:

Field testing will be performed manually and functional tests will be written in detail.

Test objectives:

1. Functional Verification: Ensure that the system functions according to the specified requirements, validating that all features work as intended.
2. Performance Assessment: Evaluate the system's performance under various load conditions to ensure it meets performance benchmarks, including response time and scalability.
3. Integration Validation: Verify that all components of the system work together correctly, ensuring seamless integration of modules and external systems.
4. Security Testing: Identify vulnerabilities and ensure that the system protects data and maintains functionality in the face of malicious attacks.
5. Usability Evaluation: Assess the system's user interface and user experience, ensuring it is intuitive and meets user expectations for ease of use.
6. Compatibility Testing: Confirm that the system operates correctly across different devices, browsers, and operating systems to ensure broad accessibility.
7. Error Handling: Validate that the system gracefully handles errors, providing meaningful messages and recovering without data loss or crashes.
8. Data Integrity Verification: Ensure that data is processed accurately and remains consistent across different operations and transactions within the system.
9. Regression Testing: Verify that new updates or changes do not adversely affect existing functionalities, maintaining overall system stability.
10. Compliance Check: Ensure that the system meets relevant regulatory and industry standards, adhering to necessary guidelines and legal requirements.

CHAPTER 9

CODING

app.py:

```
from flask import Flask, request, jsonify

app = Flask(__name__)

# Dictionary to store barcode numbers and their corresponding balances
barcode_balances = {}
common_barcode = None

@app.route('/pay', methods=['POST'])
def process_payment():
    global common_barcode
    if common_barcode is None:
        return jsonify({'error': 'No barcode scanned yet'}), 400
    data = request.get_json()
    amount = int(data.get('amount'))
    if common_barcode in barcode_balances:
        barcode_balances[common_barcode] += amount
        # return jsonify({'message': f'Payment of {amount} for {common_barcode} processed successfully'})
        return jsonify({'message': f'Payment of Rs.{amount} for {common_barcode} processed successfully'})
    else:
        return jsonify({'error': f'Invalid barcode {common_barcode}'}), 400

@app.route('/total-amount', methods=['GET'])
def total_amount():
    global barcode_balances
    total_amount = 0
```

```

try:
    # Iterate over registered barcodes and sum their balances
    for barcode in barcode_balances:
        total_amount += barcode_balances[barcode]
    return jsonify({'totalAmount': total_amount})
except Exception as e:
    return jsonify({'error': str(e)})

@app.route('/balances', methods=['GET'])
def get_balances():
    global barcode_balances
    return jsonify(barcode_balances)

@app.route('/barcode', methods=['POST'])
def handle_barcode():
    global common_barcode

    data = request.get_json()
    if not data or 'product' not in data:
        return jsonify({'error': 'Invalid request payload'}), 400
    barcode = data.get('product')
    common_barcode = barcode

    if barcode not in barcode_balances:
        barcode_balances[barcode] = 0
    return 'Barcode data received and processed successfully'

if __name__ == '__main__':
    app.run(debug=True)

```

server.py:

```
import requests

# Function to fetch total amount from Flask backend

def fetch_total_amount():
    try:
        response = requests.get('http://localhost:5000/total-amount')
        response.raise_for_status() # Raise an error for bad status codes (4xx or 5xx)
        # total_amount = response.json()['total_amount']
        total_amount = response.json()['totalAmount']
        print(f'Total Amount: {total_amount}')
    except requests.exceptions.HTTPError as http_err:
        print(f'HTTP error occurred: {http_err}')
    except requests.exceptions.RequestException as req_err:
        print(f'Request error occurred: {req_err}')
    except ValueError as val_err:
        print(f'Error decoding JSON: {val_err}')
    except Exception as e:
        print(f'Error fetching total amount: {e}')

# Call the function to fetch and display the total amount

def fetch_balances():
    try:
        response = requests.get('http://localhost:5000/balances')
        response.raise_for_status() # Raise an error for bad status codes (4xx or 5xx)
        barcode_balances = response.json()
```

```

# Process the retrieved balances

for barcode, balance in barcode_balances.items():
    print(fID No.: {barcode}, Balance: {balance}')
except requests.exceptions.HTTPError as http_err:
    print(fHTTP error occurred: {http_err}')
except requests.exceptions.RequestException as req_err:
    print(fRequest error occurred: {req_err}')
except ValueError as val_err:
    print(fError decoding JSON: {val_err}')
except Exception as e:
    print(fError fetching balances: {e}')

# Call the function to fetch and display balances

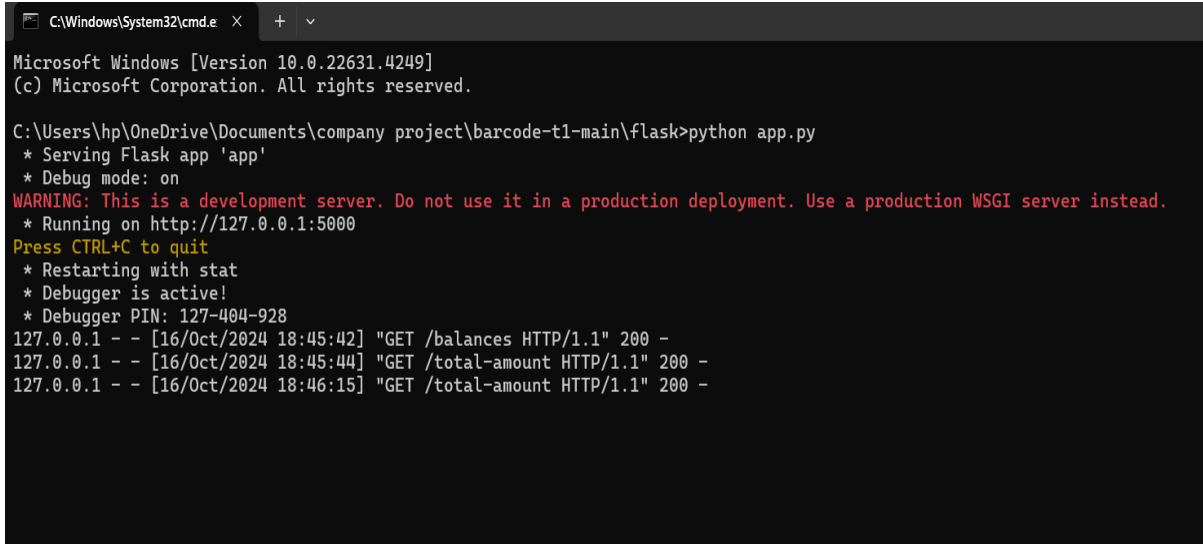
fetch_balances()
fetch_total_amount()

```

CHAPTER 10

SCREENSHOTS

- Starting app.py on the Command Prompt.



```
C:\Windows\System32\cmd.e X + v
Microsoft Windows [Version 10.0.22631.4249]
(c) Microsoft Corporation. All rights reserved.

C:\Users\hp\OneDrive\Documents\company project\barcode-t1-main\flask>python app.py
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 127-404-928
127.0.0.1 - - [16/Oct/2024 18:45:42] "GET /balances HTTP/1.1" 200 -
127.0.0.1 - - [16/Oct/2024 18:45:44] "GET /total-amount HTTP/1.1" 200 -
127.0.0.1 - - [16/Oct/2024 18:46:15] "GET /total-amount HTTP/1.1" 200 -
```

- Starting server.py on the other Command Prompt.



```
C:\Windows\System32\cmd.e X + v
Microsoft Windows [Version 10.0.22631.4249]
(c) Microsoft Corporation. All rights reserved.

C:\Users\hp\OneDrive\Documents\company project\barcode-t1-main\flask>python server.py
Total Amount: 0

C:\Users\hp\OneDrive\Documents\company project\barcode-t1-main\flask>
```

- Starting electron software using Command Prompt.
- To start -> npm start

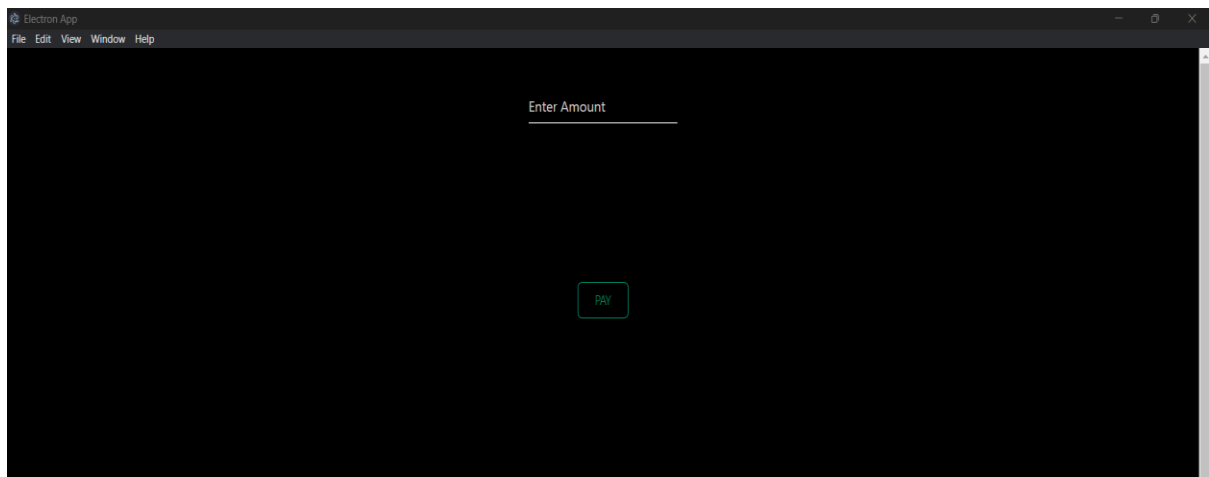
```
C:\Windows\system32\cmd.e: X + v
Microsoft Windows [Version 10.0.22631.4249]
(c) Microsoft Corporation. All rights reserved.

C:\Users\hp\OneDrive\Documents\company project\barcode-t2-main\barcode-t2-main>npm start

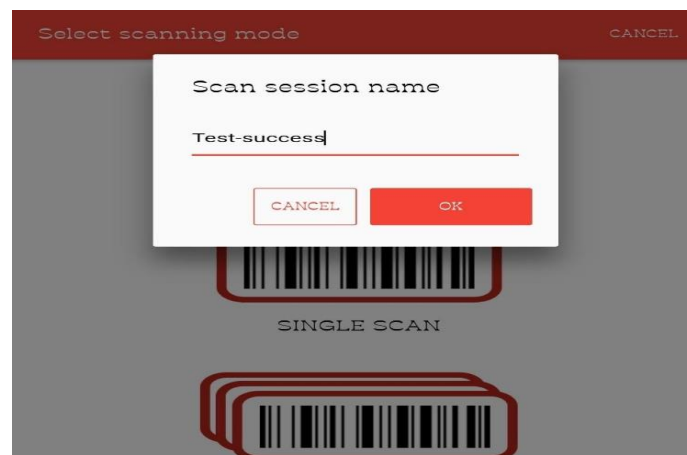
> test01@1.0.0 start
> electron .

|
```

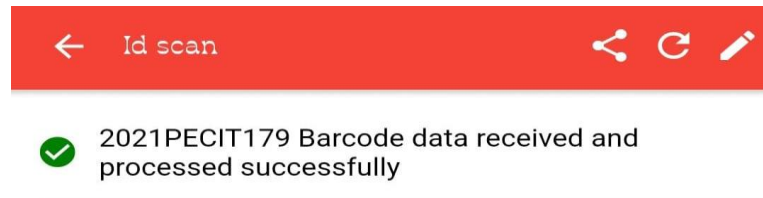
- Now, The Electron software opened and ready to process.



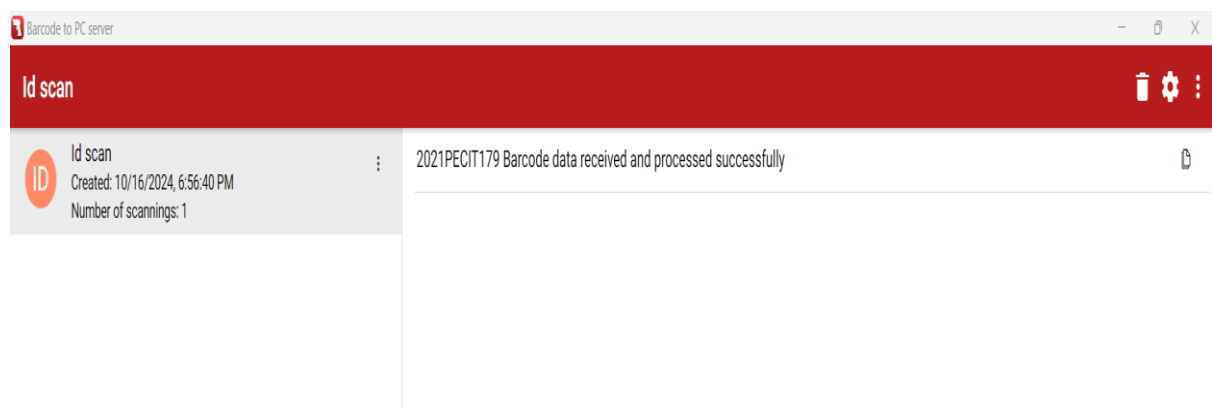
- Open Barcode to PC app on Phone and Laptop/Server.
- By Scanning the QR code connect the app on Phone with Laptop/Server.
- After open the app, click the Single Scan on phone and give the name as we want.



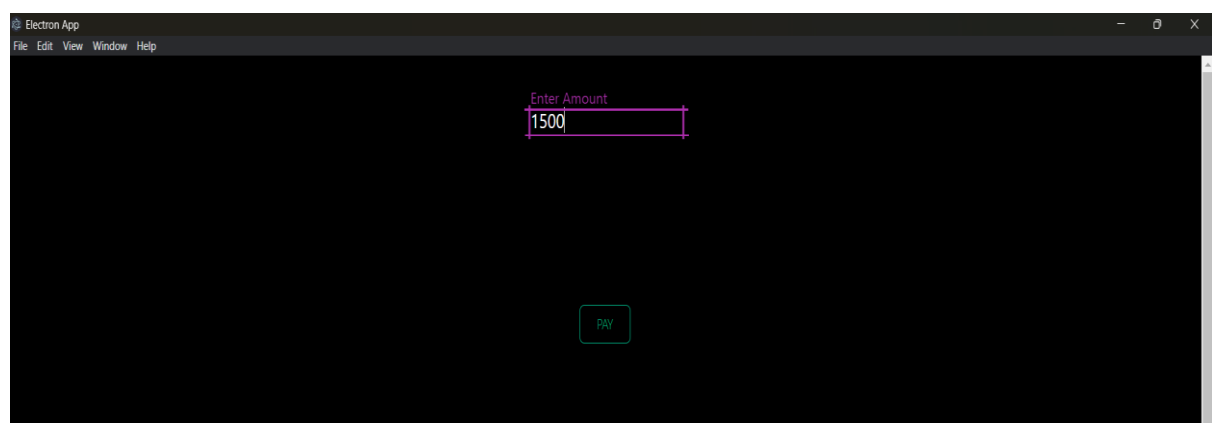
- Now Scan the barcode from ID card.
- Now the Barcode data received and processed successfully on both Phone and Laptop.
- In Phone,



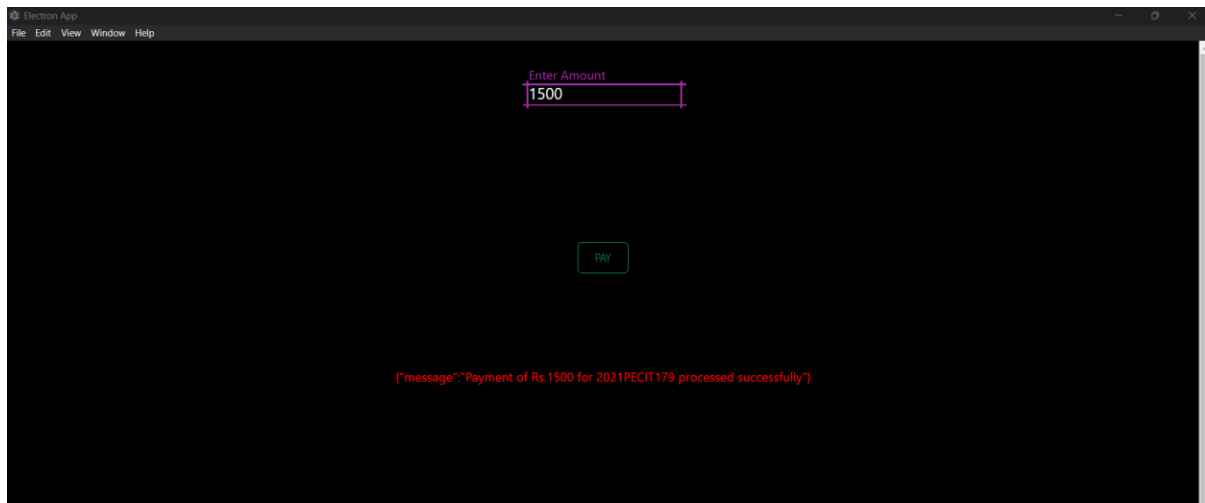
- In Server,



- After this, provide the amount in the electron app which should be paid.
- Click pay.



- After this, it sent to the server and stored in the database.



- Now, the final output -> Amount which need to be paid is stored in the ID name.

```
C:\Windows\System32\cmd.exe X + v
Microsoft Windows [Version 10.0.22631.4249]
(c) Microsoft Corporation. All rights reserved.

C:\Users\hp\OneDrive\Documents\company project\barcode-t1-main\flask>python server.py
Total Amount: 0

C:\Users\hp\OneDrive\Documents\company project\barcode-t1-main\flask>python server.py
ID No.: 2021PECIT179, Balance: 1500
Total Amount: 1500

C:\Users\hp\OneDrive\Documents\company project\barcode-t1-main\flask>
```

CHAPTER 11

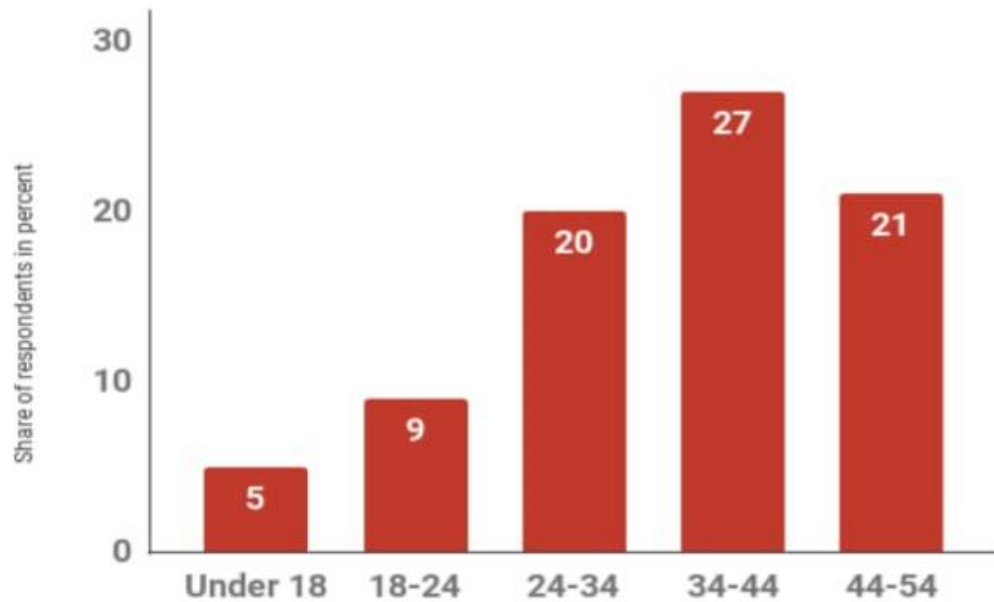
CONCLUSION

The ID Barcode Payment system developed using Python Flask and Electron offers a robust and efficient solution for modern payment processing. By leveraging the capabilities of Flask for backend development and Electron for creating a user-friendly desktop application, we have successfully integrated barcode scanning functionalities to streamline transactions. This system not only enhances the user experience by simplifying payment procedures but also ensures security and reliability through careful implementation of best practices in coding and data management. The modular design allows for easy updates and scalability, making it suitable for various applications across different industries. Future enhancements could include the incorporation of additional payment methods, improved user interface features, and real-time transaction analytics. Overall, this project demonstrates the potential of combining web technologies with desktop applications to create innovative solutions in the payment processing landscape. We hope this documentation serves as a valuable resource for developers interested in building similar systems and encourages further exploration into the integration of barcode technology in financial applications.

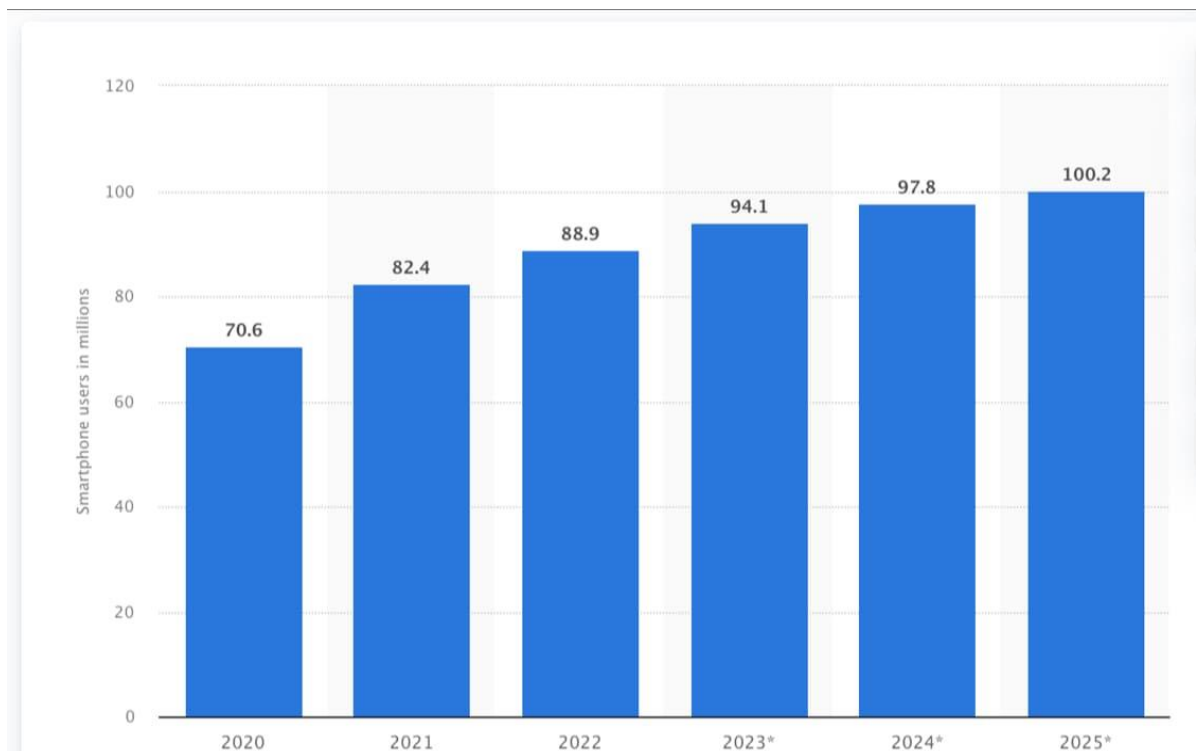
CHAPTER 12

APPENDICES

AGE Vs QR CODE:



PEOPLE USING BARCODE Vs YEAR:



CHAPTER 13

REFERENCES

- [1] Praveen Chandrahas, Deepti Kumar, Ramya Karthik, Timothy Gonsalvis, Ashok Jhunjhunwala and Gaurav Raina “ Mobile Payment Architectures for India”, National Conference on Communications,2010.
- [2] L. McCathie, “The advantages and disadvantages of barcodes and radio frequency identification in supply chain management”. 2004 <http://www.ro.uow.edu.au/thesesinfo/9/>
- [3] J. Gao, L. Prakash, and R. Jagatesan, “Understanding 2D-BarCode Technology and Applications in M-Commerce - Design and Implementation of A 2D Barcode Processing Solution,” Computer Software and Applications Conference, 2007. COMPSAC 2007. 31st Annual International, 2007, pp. 49-56.
- [4] H. Kato and K. Tan, “First read rate analysis of 2Dbarcodes for camera phone applications as a ubiquitous computing tool.”TENCON 2007 - 2007 IEEE Region 10 Conference, 2007.
- [5] Mobile Payment Forum www.mpf.org
- [6] Innopay, Mobile Payments, 2010 -
http://admin.nacha.org/userfiles/File/The_Internet_Council/Resources/Mobile%20payments%202010%20-%20Innopay.pdf
- [7] Y. Lin, M. Chang, and H. Rao, “Mobile prepaid phone services”, IEEE Personal Communications, 7(3): 6-14, June 2000.
- [8] Z. Huang, and K. Chen, “Electronic Payment in Mobile Environment”, In Proceedings of 13th International Workshop on Database and Expert Systems Applications (DEXA'02) September 02 - 06, 2002. Aix-en-Provence, France.
- [9] Smart Card Alliance Payments Council, the Mobile Payments and NFC Landscape: A U.S. Perspective September 2011.
- [10] Smart Card Alliance Payments Council Proximity Mobile Payments: Leveraging NFC and the Contactless Financial Payments Infrastructure.
- [11] Jan Ondrus, “Mobile Payments: A Tool Kit for A Better Understanding Of The Market”.
- [12] Lecture note of Intensive seminar on m-commerce Mar, 1st to 5th, 2004.
- [13] Stamatis Karnouskos, Fraunhofer Fokus, “Mobile Payment: A Journey through Existing Procedures and Standardization Initiatives”. IEEE Communications. Volume 6, No. 4, 2004.
- [14] Proximity Mobile Payments Business Scenarios: Research Report on Stakeholder Perspectives. A Smart Card Alliance Contactless Payments Council White Paper. July 2008.
- [15] Jerry Gao, Vijay Kulkarni, Himanshu Ranavat, Lee Chang, Hsing Mei, “A 2D Barcode-Based Mobile Payment System”. IEEE Third International Conference on Multimedia and Ubiquitous Engineering 2009.

- [16] J. Krasner, Embedded Market Forecasters and American Technology International Inc, "Using Elliptic Curve Cryptography (ECC) for Enhanced Embedded Security Financial Advantages of ECC over RSA or Diffie-Hellman (DH)," White Paper Series, 2004.
- [17] Certicom Research, "Standards for Efficient Cryptography -SEC 1: Elliptic Curve Cryptography", September 20, 2000.
- [18] J. McCaffrey, "Encrypt It Keep Your Data Secure with the New Advanced Encryption Standard," Microsoft MSDN Magazine, November 2003.
- [19] D. Hankerson, A. Menezes, and S. Vanstone, "Guide to Elliptic Curve Cryptography", New York: Springer, 2004, pp. 184 - 190.
- [20] N. Jobanputra, V. Kulkarni, D. Rao, and Jerry Gao, "Emerging Security Technologies for Mobile User Accesses", Accepted by The electronic Journal on E-Commerce Tools and Applications (eJETA), 2008.
- [21] Jerry Zeyu Gao., Jacky,Cai,Min Li, and Sunitha Magadi Venkateshi, "Wireless Payment – Opportunities Challenges and Solutions". Published by High Technology Letters,Vol12 ISSN 1006- 6748,2006.
- [22] Antovski and MGusev, "M-Payments", Proceedings of 25th International Conference Information Technology Interfaces, 2003(ITI'03).